Planet–disc interactions with discontinuous Galerkin methods using GPUs

Velasco Romero, David A ; Han Veiga, Maria ; Teyssier, Romain ; Masset, Frédéric S

Abstract: We present a two-dimensional Cartesian code based on high-order discontinuous Galerkin methods, implemented to run in parallel over multiple graphics processing units. A simple planet–disc setup is used to compare the behaviour of our code against the behaviour found using the FARGO3D code with a polar mesh. We make use of the time dependence of the torque exerted by the disc on the planet as a mean to quantify the numerical viscosity of the code. We find that the numerical viscosity of the Keplerian flow can be as low as a few 10−8r2Ω, r and Ω being respectively the local orbital radius and frequency, for fifth-order schemes and resolution of 10−2r. Although for a single disc problem a solution of low numerical viscosity can be obtained at lower computational cost with FARGO3D (which is nearly an order of magnitude faster than a fifth-order method), discontinuous Galerkin methods appear promising to obtain solutions of low numerical viscosity in more complex situations where the flow cannot be captured on a polar or spherical mesh concentric with the disc.

DOI: https://doi.org/10.1093/mnras/sty1192

Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **478,** 1855–1865 (2018)
Advance Access publication 2018 May 5

doi:10.1093/mnras/sty1192

Planet–disc interactions with discontinuous Galerkin methods using GPUs

David A. Velasco Romero,[1,2,3]★ Maria Han Veiga,[2,4] Romain Teyssier[2] and Frédéric S. Masset[3]

[1]*Universidad Autónoma del Estado de Morelos, Av. Universidad s/n, 62210 Cuernavaca, Mor., Mexico*
[2]*Institute of Computational Science, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland*
[3]*Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México, Av. Universidad s/n, 62210 Cuernavaca, Mor., Mexico*
[4]*Institute of Mathematics, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland*

Accepted 2018 May 2. Received 2018 April 27; in original form 2018 March 6

ABSTRACT

We present a two-dimensional Cartesian code based on high-order discontinuous Galerkin methods, implemented to run in parallel over multiple graphics processing units. A simple planet–disc setup is used to compare the behaviour of our code against the behaviour found using the FARGO3D code with a polar mesh. We make use of the time dependence of the torque exerted by the disc on the planet as a mean to quantify the numerical viscosity of the code. We find that the numerical viscosity of the Keplerian flow can be as low as a few $10^{-8}r^2\Omega$, r and Ω being respectively the local orbital radius and frequency, for fifth-order schemes and resolution of $\sim 10^{-2}r$. Although for a single disc problem a solution of low numerical viscosity can be obtained at lower computational cost with FARGO3D (which is nearly an order of magnitude faster than a fifth-order method), discontinuous Galerkin methods appear promising to obtain solutions of low numerical viscosity in more complex situations where the flow cannot be captured on a polar or spherical mesh concentric with the disc.

Key words: hydrodynamics – methods: numerical – planet–disc interactions – protoplanetary discs.

1 INTRODUCTION

The discovery of exoplanetary systems at an ever increasing pace has triggered a lot of theoretical works to understand and account for their extraordinary diversity. A significant fraction of these studies has been undertaken through intensive computational simulations in which protoplanets grow and gravitationally interact with their parent disc. The more common practice for simulations of planet–disc interactions is through grid-based codes. Among the plethora of such codes used for studies of planet–disc interactions we can cite Athena (e.g. Zhu et al. 2014), DISCO (Duffell 2016), FARGO and FARGO3D (Masset 2000; Benítez-Llambay & Masset 2016), NIRVANA (D'Angelo, Kley & Henning 2003), PENCIL (Lyra et al. 2009), PEnGUIn (Fung, Shi & Chiang 2014), PLUTO (Mignone et al. 2012), and RODEO (Paardekooper & Mellema 2006). Some of these codes are relatively new, while others have been used for over a decade. The properties and performance of the latter have been studied by de Val-Borro et al. (2006) in a code comparison project dedicated to planet–disc interactions. By far the most common meshes are polar meshes centred on the primary (in two dimensions) or cylindrical or spherical meshes (also centred on the primary, and coplanar with the disc) for three-dimensional simulations. There are very few exceptions to this, such as the studies of Pepliński, Artymowicz & Mellema (2008), who performed short-term simulations of the fast migration of giant planets. Cylindrical or spherical meshes are naturally adapted to the geometry of the problem at hand, and result in much smaller numerical viscosity of the disc's flow than their Cartesian counterpart, for a given scheme and cell size. On the other hand planet–disc interactions are very sensitive to the disc's viscosity, be it through the saturation of the corotation torque in the low-mass regime (Masset 2001; Masset & Casoli 2010; Paardekooper, Baruteau & Kley 2011) or through the gap opening processes for giant planets (Lin & Papaloizou 1986; Crida, Morbidelli & Masset 2006; Fung, Shi & Chiang 2014). There is a growing body of evidence that protoplanetary discs have a low effective viscosity, if any at all. The inclusion of non-ideal MHD effects in theoretical models of protoplanetary discs leads to a qualitatively different picture from earlier models, and suggest that the flow is laminar over most of the disc (Bai & Stone 2013; Lesur, Kunz & Fromang 2014), while attempts of detection of turbulent motion in nearby protoplanetary discs lead to ever decreasing upper limits (e.g. Flaherty et al. 2018). Studies of planet–disc interactions should therefore be undertaken with schemes of very low numerical viscosity. Not all numerical studies of protoplanetary discs or environments can be done on cylindrical or spherical

★ E-mail: david.velasco@icf.unam.mx

meshes, however. As they grow in complexity and realism, they may be better done on Cartesian meshes with AMR (Lichtenberg & Schleicher 2015; Hennebelle, Lesur & Fromang 2017). This can also happen if several discs are considered at the same time, such as the circumstellar discs of a multiple star. Under such circumstances, reaching the very low levels of viscosity required to capture correctly the interaction between the disc and its forming planets may prove challenging. Recently, Schaal et al. (2015) presented an implementation of discontinuous Galerkin (DG) schemes aimed at describing astrophysical flows. The high order of the solutions provided by these schemes suggests that they may be able to capture differentially rotating discs with a very low viscosity. Besides, these schemes present the interesting property that they conserve angular momentum to machine accuracy in the parts of the flow where no limiting occurs. This property is highly desirable for long-term simulations of planet–disc interactions, where most of the planet's drift can be accounted for by an exchange of angular momentum between the planet and its coorbital region: a spurious change of the angular momentum of the latter may induce an erroneous migration rate of the former. Since DG methods are compute intensive and have a small stencil, they are well suited to massive multithreaded platforms such as *graphics processing units* (GPUs).

For all the reasons explained above, we have implemented a two-dimensional, Cartesian version of DG schemes on GPUs, and evaluated their properties on Keplerian flows with embedded, intermediate mass planets. Although an implementation of DG schemes in cylindrical or spherical coordinates would be feasible, we regard our present implementation as a proof of concept in the least favourable case. As we shall see, we are able to obtain very small numerical viscosities even in the case of a Cartesian mesh, which suggests that better results would be attainable for coordinate systems fitted to the geometry of the flow. Our paper is organized as follows: in Section 2, we recall the main features of DG schemes, and provide some details about our implementation in Section 3. We then check our code's behaviour and convergence properties on standard tests in Section 4 and we present our results for the problem of a planet embedded in a protoplanetary disc in Section 5. We use the time behaviour of the corotation torque as a diagnostic to evaluate the effective viscosity of the disc. At finite viscosity, this torque tends towards a finite, constant value which depends on the effective viscosity, whereas it oscillates and tends to zero in inviscid discs. The asymptotic torque therefore constitutes an accurate measure of the disc's effective viscosity, albeit somehow indirect. Note that although our method can accurately determine the effective numerical viscosity of a given scheme, the exact value may differ if another method is used. We finally draw our conclusions in Section 6.

2 PRINCIPLES OF DISCONTINUOUS GALERKIN SCHEMES

2.1 Governing equations

The Euler equations describe how the velocity, pressure, and density of a moving fluid are related under the influence of a source term. They form an n-dimensional system of hyperbolic partial differential equations that can be written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{n} \frac{\partial}{\partial x_i} \boldsymbol{f}_i(u) = \boldsymbol{S}(\boldsymbol{u}, \boldsymbol{x}), \tag{1}$$

where

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \rho v_x & \rho v_y \\ \rho v_x^2 + p & \rho v_x v_y \\ \rho v_x v_y & \rho v_y^2 + p \\ (E+p)v_x & (E+p)v_y \end{pmatrix},$$

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ -\rho \frac{\partial}{\partial x} \Phi \\ -\rho \frac{\partial}{\partial y} \Phi \\ -\rho \boldsymbol{v} \cdot \nabla \Phi \end{pmatrix}$$

for a two-dimensional flow under the influence of a gravitational source term with potential Φ. Here $\boldsymbol{f}$ represents the matrix of fluxes $(\boldsymbol{f}_x, \boldsymbol{f}_y)$ and $\boldsymbol{S}$ the source term.

The unknown quantities are density ρ, velocity $\boldsymbol{v} = (v_x, v_y)$, pressure p, and total energy E. The total energy can be expressed in terms of the density of internal energy e and kinetic energy of the fluid, $E = e + \frac{1}{2}\rho \boldsymbol{v} \cdot \boldsymbol{v}$. For an ideal gas, the system is closed with the equation of state

$$p = e(\gamma - 1), \tag{2}$$

where γ denotes the adiabatic index.

2.2 Discontinuous Galerkin method

We follow the method formulated by Cockburn & Shu (1998), which we summarize below for a two-dimensional scalar conservation law defined on a Cartesian grid.

Consider a regular two-dimensional domain $\Omega \in \mathbb{R}^2$, approximated by non-overlapping rectangular elements

$$K_{i,j} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}],$$

where (i, j) indexes the rectangles. Furthermore, consider the local space $V(K)$ given by the set of two-dimensional polynomials with degree of at most N_p in x and y. We denote $\{\phi_i\}_{i=0}^{N_p}$ to be the set of polynomial basis of the local space $V(K)$.

For every rectangle $K_{i,j}$, the local solution is expressed as[1]

$$u_h^K(\boldsymbol{x}, t) = \sum_{i=0}^{N_p} \hat{u}_i^K(t) \phi_i(\boldsymbol{x}).$$

In this work we use a *modal representation* of the solution. This means that the numerical solution in element K is represented by the linear coefficients of the basic functions $\hat{u}_i^K(t)$ for $i = 0, ..., N_p$. In particular, Legendre polynomials are chosen as the polynomial basis because they are orthogonal to each other, i.e. $\int \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \delta_{ij}$.

A modal coefficient $\hat{u}_i^K(t)$ is obtained with the L^2 projection of the solution $u(x)$ restricted to element K on the orthogonal basis vector $\phi_i(\boldsymbol{x})$:

$$\hat{u}_i^K(t) = \int_K u(\boldsymbol{x}, t) \phi_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}.$$

The pointwise values of the solution (nodal values) $u_h^K(\boldsymbol{x}, t)$ can be recovered by

$$u_h^K(\boldsymbol{x}, t) = \sum_{i=0}^{N_p} \hat{u}_i^K(t) \phi_i(\boldsymbol{x}).$$

[1] We drop the rectangle indices i, j when it is not important to specify them.

Finally, the scalar global solution $u(\boldsymbol{x}, t)$ is given by *stitching* together the local solutions defined in each local subspace $V(K)$, which is formally expressed as a direct sum (denoted with $\oplus$):

$$u(\boldsymbol{x}, t) \approx u_h(\boldsymbol{x}, t) = \oplus_{K \in \Omega_h} u_h^K(\boldsymbol{x}, t).$$

The extension to a system of equations is done by repeating the treatment described above for each variable in the vector solution.

2.2.1 *Space discretization*

Using the notation above, we discretize equation (1) in space using the DG method. For each time t, the approximate solution $u_h(\boldsymbol{x}, t)$ is sought in the finite element space of discontinuous functions. The weak formulation of equation (1) is attained by multiplying the equation by a smooth test function $v(\boldsymbol{x})$, integrating over a control volume K and applying the divergence theorem:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K u(\boldsymbol{x}, t)v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \sum_{e\in\partial K}\int_e f(u(\boldsymbol{x}, t))\cdot n_{e,K}\, v(x)\,\mathrm{d}\Gamma - \int_K f(u(\boldsymbol{x}, t))\cdot\nabla v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_K S(u)v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}.$$

Here $n_{e,K}$ denotes the outward unit normal to the edge e.

The exact solution is replaced with the approximate solution $u_h(\boldsymbol{x})$, the test function $v(\boldsymbol{x})$ by $v_h(\boldsymbol{x})$ and the integrals from the weak formulation are replaced by a suitable quadrature, yielding the semidiscrete formulation of the DG method, written as

$$\begin{aligned} u_h(t=0) &= P_{V_h}(u_0) \\ \frac{\mathrm{d}}{\mathrm{d}t}\int_K u_h(\boldsymbol{x}, t)v_h(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} &= -\sum_{e\in\partial K}\sum_{i=0}^{L} h_{e,K}(\boldsymbol{x}_i, t)v_h(\boldsymbol{x}_i)w_i|e| \\ &\quad + \sum_{j=0}^{M} f(u(\boldsymbol{x}_j, t))\cdot\nabla v_h(\boldsymbol{x}_j)w_j|K| \\ &\quad + \sum_{j=0}^{M} S(u(\boldsymbol{x}_j, t))v_h(\boldsymbol{x}_j)w_j|K| \\ &\quad \forall v_h(\boldsymbol{x}) \in V(K)\forall K \in \Omega_h, \end{aligned} \tag{3}$$

where $P_{V_h}(\cdot)$ denotes the L^2 projection of the initial data $u_0(\boldsymbol{x})$ into the space of finite elements V_h, $\{(\boldsymbol{x}_i, w_i)\}_{i=0}^{L,M}$ are sets of Gauss–Legendre quadrature points (with their respective weights w) with different number of points L and M, for the edge and volume integrals, $|e|$ the length the edge and $|K|$ the area of the control volume. Furthermore, $f(u(\boldsymbol{x}, t))\cdot n_{e,K}$ is replaced by $h_{e,K}(x_i, t)$, the numerical flux, which determines a unique solution at the interface shared between neighbouring elements.

2.2.2 *Time discretization*

Because we choose our local solution space to be a set of orthogonal polynomials, we have an expression for the evolution of each mode $\hat{u}_i^K$ independently of the other modes. The semidiscrete form (3) reduces the partial differential equation to an ordinary differential equation of the form:

$$\frac{\mathrm{d}}{\mathrm{d}t}u_h = \mathcal{L}(u),$$

where $\mathcal{L}$ denotes the right-hand side of (3). A strong stability preserving (SSP) Runge–Kutta (RK) time discretization is used (Gottlieb & Shu 1998). The time marching algorithm is detailed in Algorithm 1.

The coefficients for a_{ij}, b_i, and c_i can be found in Appendix A.

Data: $w_h^0 = P_{V_h}(w_0)$
Result: w_h^{n+1}
for $n = 0, \ldots N-1$ **do**
  $w_h^{(0)} = w_h^n$;
  $h = \Delta t$;
  for $i = 0, \ldots k$ **do**
    $k_i = \mathcal{L}(t^n + c_i\cdot h, y_i + h(a_{i,1}k_1 + \ldots + a_{i,i-1}k_{i-1}))$;
    $w_h^{(i+1)} = w_h^{(i)} + h\sum_{j=1}^{i} b_j k_j$;
  end
  $w_h^{n+1} = w_h^{(k+1)}$;
end

Algorithm 1. TVD RK time marching algorithm.

2.3 Time-step

When using an explicit time integrator, the time-step has to fulfil a Courant–Friedrich–Lewy (CFL) condition to achieve numerical stability. The time-step Δt^K of the cell K is calculated as Cockburn & Shu (1998).

$$\Delta t^K = \frac{C}{2N_p+1}\left(\sum_{i=1}^{d}\frac{|v_i^K| + c_s^K}{\Delta x_i^K}\right)^{-1},$$

where $c_s = \sqrt{\gamma p/\rho}$ is the sound speed, v_i^K is the ith component of the velocity average at cell K, Δx_i^K the mesh width in the ith dimension.

2.4 Solution limiters

It is known that non-linear equations can develop discontinuities at finite time and that non-physical oscillations develop in the numerical solution in the presence of discontinuities. These, in turn, reduce the pointwise accuracy of the method, lead to loss of convergence near the discontinuity and to the appearance of artificial and persistent oscillations near the discontinuity point (Hesthaven & Warburton 2007). Furthermore, for physical systems, it is of interest to have the solution fulfilling certain constraints, such as positivity or boundedness (e.g. positive pressure and density). To stabilize the solution, limiters can be used. These, in turn, will affect the quality of the numerical solution.

In this work, we make use of a positivity preserving limiter (Zhang & Shu 2010) to guarantee that the pressure and density remain positive. When using this limiter, there is a further restriction on the time-step, which includes the weight of the first Gauss Lobatto quadrature node, denoted as w_1, appropriate for the limiter for a N_pth order approximation:

$$\Delta t^K = C\min\left(\frac{1}{2N_p+1}, \frac{w_1}{2}\right)\left(\sum_{i=1}^{d}\frac{|v_i^K| + c_s^K}{\Delta x_i^K}\right)^{-1}.$$

3 IMPLEMENTATION

The availability of computational resources such as GPUs has brought renewed interest in compute intensive methods, in which performance is bound by sheer computation rather than by memory access. The DG methods having a small stencil and being compute intensive on most platforms fit these requirements. Here we present a two-dimensional Cartesian implementation of the DG method on GPUs, using CUDA and MPI.

3.1 Overview of the algorithm

The succession of steps of our implementation is as follows:

(i) Initial conditions

(a) Initialize nodes of primitive variables
(b) Convert to nodes of conservative variables
(c) Integrate to modes of conservative variables

(ii) CFL condition: Find global time-step
(iii) Runge–Kuta sub-stepping:

(a) Compute: volume fluxes, face fluxes, source terms
(b) Compute modal update
(c) Apply boundary conditions
(d) Apply limiters

(iv) If $t = t_{\rm output}$: Copy modes to CPU and output them
(v) If $t < T_{\rm end}$: Return to step (ii)
(vi) End simulation

The evaluation of the limiting time-step is done from the zeroth-order modes, which are the average values for each element. The reduction to obtain the global time-step is done over a single block of threads making use of shared memory. We also make use of the GPU's so-called constant memory to store quadrature values and their respective weights as well as the Legendre polynomials evaluated at these quadrature points.

In order to make an implementation capable of running in parallel over several GPUs there is the need to divide the initial domain, in our case building a sub-domain per GPU. The amalgamation of these sub-domains is then done via *Boundary Conditions*, this allows us to design all the other parts of the code as if each sub-domain were an individual domain with $n_x n_y$ active cells and just one layer of inactive cells or 'ghost' cells per side. The information to be communicated consist of the modal values for the conservative quantities. In our implementation (mixing FORTRAN and CUDA) we did not make use of CUDA-aware MPI instructions to perform device to device memory transfers, therefore we still have room available to increase the performance of our code on multi-GPUs platforms.

From now on we will use the expression 'degree of freedom' to refer to a single element of resolution, so that the number of degrees of freedom is $n_x \times n_y$ in FARGO3D and $n_x \times n_y \times m^2$ for a DG scheme of order m (note that the number of values that can be set independently to specify a given configuration is four times larger, since we can specify the surface density, the pressure and the two components of the velocity for each element of resolution).

In our present implementation, all our fields consist of linear arrays, and all our kernels are one-dimensional. The mapping of threads to modes was chosen so as to facilitate memory access without enforcing coalesced transactions: we have one matrix per mode, resulting in $m \times m$ matrices of size $n_x \times n_y$ rather than a unique matrix of size $(m \times nx) \times (m \times n_y)$. Previous experimentation with the automatic data management on the GPU with FARGO3D (see Benítez-Llambay & Masset 2016) has shown that using pitched memory for multidimensional arrays led to little improvement, if any at all. This is likely due to the two levels of cache available on modern GPUs, as well as sophisticated transaction mechanisms with the global memory on GPUs with recent compute capabilities, leveraging the requirement for alignment, which is no longer so much of a concern since compute capabilities 2.0.

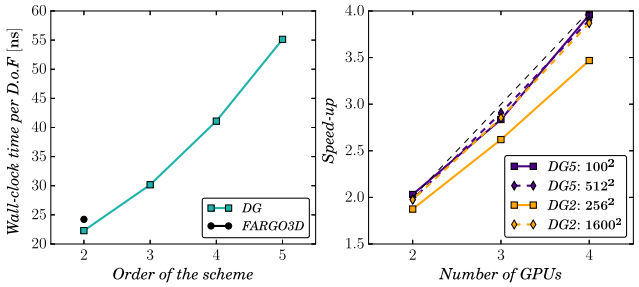

Figure 1. On the left is the average sub-step time taken per degree of freedom as a function of the method's order. On the right is the speed-up observed in the DG code as a function of the number of GPUs used. These data were gathered using NVIDIA's Tesla K20s with error control (ECC) activated.

3.2 GPU performance

With the purpose of validating our implementation of the DG method, we measured the wall-clock time of a sub-step with different resolutions and different spatial orders. We then compared these times against the ones measure for FARGO3D. Our results are shown on the left side of Fig. 1, where we plot the average execution time *per degree of freedom* against the order of the scheme. It is important to clarify that this is the wall-clock time for one sub-step, which in DG translates to one of the stages in the *RK* sub-stepping. On the right side we present strong scaling curves to show the performance of the code as a function of the number of GPUs used. We can observe that even though we do not make use of CUDA-aware MPI instructions we still have a scaling close to the optimal one. We can also see that with higher order it is possible to get closer to optimal scaling even with low resolutions. A weak scaling test showed that we obtain a 60×speed up ratio when running the code on 64 P100 GPUs.

We mention that we have developed over the past year a GPU version only of the code, so we cannot quote an accurate speed up ratio with respect to a CPU core. However, we observed with an earlier, non-optimized CPU version of the code, a speed up ratio comprised between 100 and 350 (a larger ratio is obtained at higher order of the scheme, which is likely due to the fact that higher order schemes are most compute intensive). This ratio was obtained respectively with K80 GPUs and Intel™ Xeon E5 cores.

4 TEST PROBLEMS

We present hereafter a number of standard test problems in order to validate our implementation.

4.1 Isentropic vortex

The isentropic vortex problem describes the convection of an isentropic vortex in an inviscid flow (Yee, Sandham & Djomehri 1999). The physical domain is the square $[0, 10] \times [0, 10]$, the vortex is centred at $(x_c, y_c) = (5.0, 5.0)$, $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ and the boundary conditions are periodic. The initial conditions for the

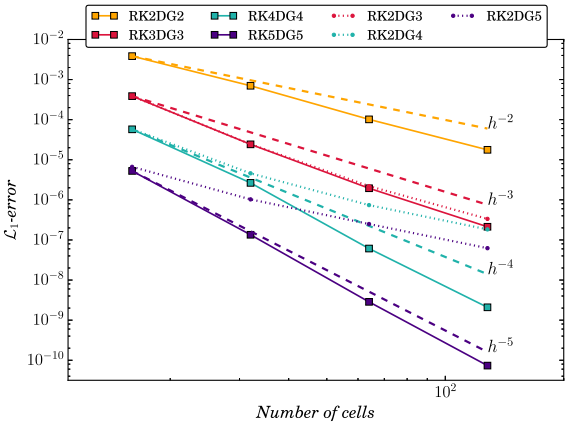

Figure 2. Convergence of the RKDG method in $\mathcal{L}_1$-norm for different spatial and time discretization orders for the isentropic vortex case.

primitive variables are

$$\rho = \left[1 - \frac{(\gamma-1)\beta^2}{8\gamma\pi^2}\exp\left(1-r^2\right)\right]^{\frac{1}{\gamma-1}},$$
$$v_x = 1 - \frac{\beta}{2\pi}\exp\left(\frac{1-r^2}{2}\right)(y-y_c),$$
$$v_y = 1 + \frac{\beta}{2\pi}\exp\left(\frac{1-r^2}{2}\right)(x-x_c),$$
$$p = \rho^\gamma,$$

for $\gamma = 1.4$, while the free stream conditions are given by

$$\rho = 1.0, \quad v_{x,\infty} = 1.0, \quad v_{y,\infty} = 1.0, \quad p = 1.0.$$

4.1.1 *Empirical convergence rate*

The empirical error estimates are calculated using the $\mathcal{L}_1$-error norm:

$$\mathcal{L}_1 = ||u_h(\boldsymbol{x}) - u(\boldsymbol{x})||_1, \quad \boldsymbol{x} \in \Omega. \tag{4}$$

It is shown by Zhang & Shu (2004) that a convergence rate of $N_p + 1$ in $\mathcal{L}_1$-norm is expected for approximate polynomial solutions of degree N_p and smooth enough solutions. This quantity is computed with a suitable numerical quadrature:

$$||u_h(\boldsymbol{x}) - u(\boldsymbol{x})||_1 \approx \sum_{K\in\Omega}\frac{1}{4}\sum_{i=0}^{N_p}\sum_{j=0}^{N_p} \mid u_h(x_i, y_j) - u(x_i, y_j) \mid w_i w_j \Delta x \Delta y. \tag{5}$$

The system is evolved until $T = 10$, i.e. until the vortex crosses the box and returns to its initial position.

As shown in Fig. 2, we observe an empirical convergence rate which is close to the expected theoretical one. We note that reducing the order of the time integration still leads to a decrease in the $\mathcal{L}_1$-norm, although the convergence rate becomes dominated by the time integration error. However, as shown in Young & Ooi (2004), it is possible to recover the right convergence rate if the CFL condition is lowered. In the practical sense, this means that for further experiments, we might be able to reduce the order of the time integration instead of matching the spatial integration order with the time integration order and still attain a low error. This is relevant, as for higher than fourth-order time integration, it is necessary to have a number larger than the desired order of Runge–Kutta sub-steps (Ruuth & Spiteri 2002), which becomes prohibitively expensive.

4.2 Gresho vortex

The Gresho vortex problem is a rotating steady solution for the inviscid Euler equations (Liska & Wendroff 2003), often used to test conservation of vorticity and angular momentum. The angular velocity v_φ depends only on the radius and the centrifugal force is balanced by the pressure gradient. The smoothing of the angular velocity profile is a measure of how well the code preserves angular momentum (Springel 2010b). The physical domain is defined by $[0, 1] \times [0, 1]$, the vortex is centred at $(x_c, y_c) = (0.5, 0.5)$ and $r = \sqrt{(x-x_c)^2 + (y-y_c)^2}$. The boundary conditions are gradient free:

$$\nabla u(\vec{x}) \cdot \vec{n}|_{\vec{x}\in\partial\Omega} = 0, \quad \text{for } u \text{ a conserved variable } \rho, v_x, v_y, p.$$

The initial conditions for the primitive variables are

$$\rho = 1.0, \quad v_x = -v_\phi\frac{(y-y_c)}{r},$$
$$v_y = v_\phi\frac{(x-x_c)}{r}, \quad p = p(r),$$

with the orbital velocity v_φ and pressure p:

$$v_\phi(r) = \begin{cases} 5r & r < 0.2 \\ 2 - 5r & 0.2 \le r < 0.4, \\ 0 & r \ge 0.4 \end{cases}$$

$$p(r) = \begin{cases} 5 + \frac{25}{2}r^2 & r < 0.2 \\ 9 - 4\log(0.2) + \frac{25}{2}r^2 - 20r + 4\log(r) & 0.2 \le r < 0.4. \\ 3 + 4\log(2) & r \ge 0.4 \end{cases}$$

The angular momentum $\vec{J}$ and vorticity $\vec{\omega} = \nabla \times \vec{v}$ can be written analytically as

$$\vec{J}(r) = \begin{cases} 5r^2 & r < 0.2 \\ 2r - 5r^2 & 0.2 \le r < 0.4, \\ 0 & r \ge 0.4 \end{cases}$$

$$\vec{\omega}(r) = \begin{cases} 10 & r < 0.2 \\ \frac{2}{r} - 10 & 0.2 \le r < 0.4. \\ 0 & r \ge 0.4 \end{cases}$$

It has been shown that the unlimited DG scheme can preserve angular momentum when choosing appropriate basis functions (Schaal et al. 2015). In Fig. 3 is shown the profile for the angular momentum at $T = 3.0$ and at $T = 50.0$ (corresponding to approximately 2.4 and 40 orbits at $r = 0.2$). We note that the angular momentum remains well captured over a longer term evolution, as expected. It has been reported (Springel 2010a; Boxi, Chao & Shusheng 2017) that the vortex breaks up for methods which are either too dissipative or unsuitable. However, the vorticity does not behave well over longer term evolution but this is not surprising as the vorticity profile is discontinuous and vorticity is measured is through higher moments of the solution.

We compare our implementation of the DG scheme with other codes which were benchmarked (Liska & Wendroff 2003) with the Gresho vortex case. Following the described setup, we evolve the flow until $T = 3$, on a mesh of size $(N_x, N_y) = (40, 40)$.

As shown in Table 1, we find that overall DG methods yield much better results than the other ones for the density (except the second order one, which yields an error comparable to that of PPM or VH1), and an error broadly similar to other methods for the vorticity (except for the fourth order DG scheme, for which the error is typically a factor of two lower than that of PPM or VH1).

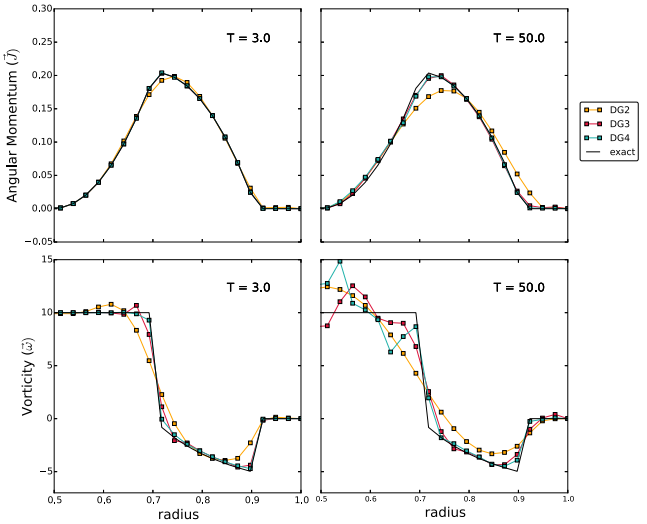

Figure 3. Angular momentum profile at time $T = 3.0$ and $T = 50.0$ for the Gresho's vortex problem for different discretization orders.

5 PROTOPLANETARY DISC WITH AN EMBEDDED PLANET

5.1 Setup

We devised a simple planet–disc setup in order to perform a comparison between our DG code and FARGO3D. The setup consists of a disc with an internal radius $r_{\rm in} = 0.4$ and an external radius $r_{\rm ex} = 1.75$, an initially uniform surface density $\Sigma_0 = 1$ and an initially uniform pressure $p_0 = 2.5 \times 10^{-3}$. A planet is on a fixed circular orbit at $r_p = 1$. Since there is no gradient of pressure and density at the planet's orbit, there is neither a gradient of entropy nor of temperature, and we therefore expect that the corotation torque acting on the planet is only the vortensity related corotation torque (e.g. Jiménez & Masset 2017, and refs. therein). The adiabatic sound speed at the planet location is therefore $c_s^{\rm adi} = \sqrt{\gamma p_0/\Sigma_0} \approx 0.059 r_p \Omega_p$ (where Ω_p is the planet's orbital frequency), while the isothermal sound speed is $c_s^{\rm iso} = 0.05 r_p \Omega_p$, which corresponds to a pressure scale length $H = 0.05 r_p$, hence the disc's aspect ratio at the planet location is $h = H/r_p = 0.05$. The planet mass is $M_p = 6.0 \times 10^{-5} M_*$, where M_* is the mass of the central object.[2] The planet's gravitational potential has a smoothing length $\epsilon_p = 0.03 r_p$. Since our frames, both in our DG codes and in FARGO3D, are centred on the star, they are not strictly inertial, the star being accelerated by the planet and the disc. This gives rise to an additional term in the gravitational potential, called the indirect term. This term is in general minute and it is not crucial for the comparison that we undertake, so we discard it hereafter in the two codes. We use the unit system in which M_* is the mass unit, r_p the length unit, and Ω_p^{-1} the time unit, which implies that in this unit system the gravitational constant G is unitary.[3] For the *DG* setup we use a square box with a side of length $4.5 r_p$ (going from $-2.25 r_p$ to $2.25 r_p$).

[2]This would translate into a 20 $M_\oplus$ planet for a central mass equal to that of the Sun.

[3]Should we take into account the indirect term, we should rather take $M_* + M_p$ as the mass unit for this statement to hold.

The fields are initialized as

$$\Sigma = \frac{\Sigma_0}{1 + f(r)}$$
$$v_x = -v_\phi \frac{y}{r}$$
$$v_y = v_\phi \frac{x}{r}$$
$$p = \frac{p_0}{1 + f(r)},$$

where we chose $f(r) = \exp\left(\frac{r - r_{\rm ex}}{r_p h}\right)$ to provide a smooth transition at the outskirts of the disc. The angular velocity v_φ that leads to rotational equilibrium with this density profile is

$$v_\phi^2 = r^2\Omega^2 + \frac{r\partial_r(c_{\rm s,iso}^2 \Sigma)}{\rho} = \Omega^2 - \frac{r c_{\rm s,iso}^2 f(r)}{r_p h[1 + f(r)]},$$

where the ratio γ of specific heats is set to 1.4. For the orbital frequency Ω we have solid rotation inside an inner limit and a Keplerian flow elsewhere:

$$\Omega^2(r) = \begin{cases} \frac{GM_*}{r_{\rm in}^3} & r \le r_{\rm in} \\ \frac{GM_*}{r^3} & r > r_{\rm in}. \end{cases}$$

The setup also includes wave-killing boundary conditions as described in de Val-Borro et al. (2006). A field Q is dampened towards its unperturbed value Q_0 every time-step according to the following prescription:

$$Q = \frac{\Delta t\, Q_0 + \tau Q}{\tau + \Delta t},$$

the damping time being

$$\tau = 2\pi \sqrt{\frac{r_d^3}{GM_*}} \times \frac{1}{R(r)},$$

where the ramp function

$$R(r) = \left(\frac{r - r_d}{r_{\rm in/ex} - r_d}\right)^2$$

is chosen to span the interval 0 to 1 with a parabolic behaviour in the zone from r_d to the boundary radius $r_{\rm in/ex}$. The damping radius r_d is chosen for the internal boundary as

$$r_d = r_{\rm in} 1.15^{3/2}$$

and for the external one as

$$r_d = r_{\rm ex} 1.15^{-3/2}.$$

The dampened fields are the density, velocities, and temperature.

5.2 Results comparison

We present the results of the Cartesian *DG* code and those of *FARGO3D* with a polar grid, both codes performing numerical simulations of the planet–disc setup. We undertook simulations with increasing resolution and order for the *DG* code. In Fig. 4 we show a comparison of the surface density map obtained with a RK2DG5 scheme and another obtained with FARGO3D. The location and contrast of the spiral wake is almost undistinguishable between the two runs as long as one stands away from the boundaries. Besides, the region where the torque originates is located relatively far from the region that limits the time-step through the CFL condition. As a consequence, the effective Courant number of this specific region is small, which results in minute differences at a given location from

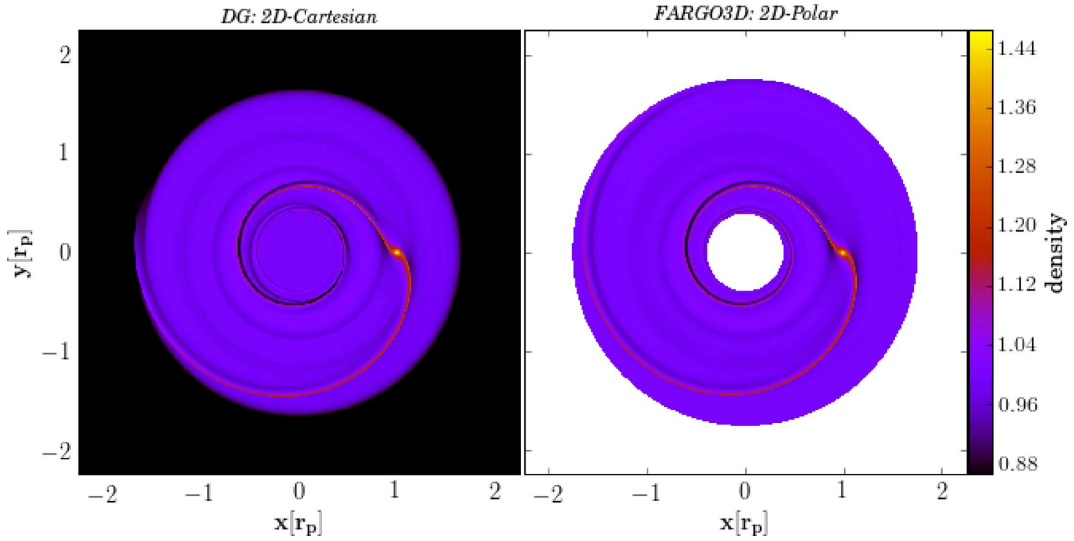

Figure 4. Snapshots of the surface density for an adiabatic disc 50 orbits after the insertion of the planet. On the left we plot results of the RK2DG5 scheme with a 640 × 640 Cartesian mesh whereas on the right we plot results of FARGO3D with a 3574 × 768 polar mesh and orbital advection.

one time-step to the next, and thus different time order schemes yield very similar results.

For the whole set of runs we monitored the specific torque exerted on to the planet position $\vec{r}_p$:

$$\vec{\Gamma} = \sum_n \vec{r}_p \times \vec{g}_n = \vec{r}_p \times \sum_n \frac{GM_n(\vec{r}_n - \vec{r}_p)}{\left[(\vec{r}_n - \vec{r}_p)^2 + \epsilon_p^2\right]^{3/2}}, \tag{6}$$

where M_n and $\vec{r}_n$ represent respectively the mass and position of cell n, whereas $\vec{g}_n$ represents the acceleration imparted by the material of cell n at the planet's location. The evaluation of this acceleration includes the planet's smoothing length ϵ_p. Given that we are considering a two-dimensional case, the only non-zero component of $\vec{\Gamma}$ is the z-component. From here on we will refer to this component as the total torque Γ.

We normalize this torque to $\Gamma_0 = \Sigma\Omega^2 r_p^4 q/h^2$ and from now on quote values of $\gamma\Gamma/\Gamma_0$.

We start by presenting in Fig. 5 the results of the *DG* code, where we show how the response depends on resolution for different orders of the scheme. As we increase the order we observe a torque exhibiting more and more the serrated behaviour expected for inviscid discs (Ward 2007).

In Fig. 6 we show the results of the DG5 scheme with RK2 and RK5 time integrators. These results can also be generalized to third and fourth spatial order schemes, for which we find that a second time order integrator yields virtually undistinguishable torque estimates. This is likely due to the fact that the horseshoe region, from which most of the torque originates, is resolved on a relatively small number of zones. As shown previously in Fig. 2, at low resolution, second order of time integrators led to errors on the norm very similar to higher order time integrators.

Fig. 7 contains the normalized total torque for different orders with similar resolution. Here the resolution for the *DG* code is taken as the cell length over the order of the approximation. The runs for *FARGO3D* were designed to have square cells at $r = r_p$, the first curve here has same cell size as the ones shown for *DG* with second, fourth, and fifth order.

We observe a behaviour resembling more closely analytical expectations for higher orders with an equal number of degrees of freedom. The torques in the top plot of Fig. 7 show a high degree of similitude between both codes, especially for the highest orders of the DG scheme. The similitude even holds if we focus on the high frequency, low amplitude components of the torque at early stages (bottom plot of Fig. 7) where the oscillations in both codes have a similar structure. Given the marked difference between the two codes (both in numerical method and mesh geometry), this strongly suggests that this behaviour is of physical origin rather than being a numerical artefact.

5.3 Estimated numerical viscosity

Horseshoe dynamics, which give rise to the corotation torque, can be essentially reduced to an advection and diffusion problem, in which the advection stems from the Keplerian flow and the diffusion comes here from the numerical scheme itself. In order to quantify the numerical viscosity of the DG method we resort to a comparison with high resolution simulations performed with the FARGO3D code using orbital advection. Since FARGO3D solves the Navier–Stokes equations, we run a set of simulations spanning kinematic viscosities ν from $10^{-9} r_p^2\Omega_p$ to $10^{-4} r_p^2\Omega_p$. We then obtain the palette of torques presented in Fig. 8, to which we can compare the results of our DG code in order to assess its numerical viscosity for a given resolution and scheme order.

Masset & Casoli (2010) find that successive maxima and minima of the corotation torque for low values of the shear viscosity are approximately in geometric sequence. We extract the ratio $\tilde{\rho}$ of this sequence using the first three extrema of our low viscosity runs (up to $\nu \approx 10^{-6} r_p^2\Omega_p$). This value of $\tilde{\rho}$ is what we use to match the physical viscosity of FARGO3D runs to the numerical viscosity of DG runs. Namely, from the values if $\tilde{\rho}$ obtained from the FARGO3D runs we build a polynomial approximation of the relation between $\tilde{\rho}$ and the viscosity ν. This relation, in turn, is used backwards to get a viscosity estimate for a given value of $\tilde{\rho}$ measured in a DG run. The estimates of the numerical viscosity are shown in Fig. 9. We stress that our method yields an accurate evaluation of the numerical viscosity of a given scheme in the very specific problem of horseshoe dynamics. Should an observable other than the torque be used to infer the numerical viscosity of a scheme (such as the minimum density in a gap opening situation, for instance), a different value

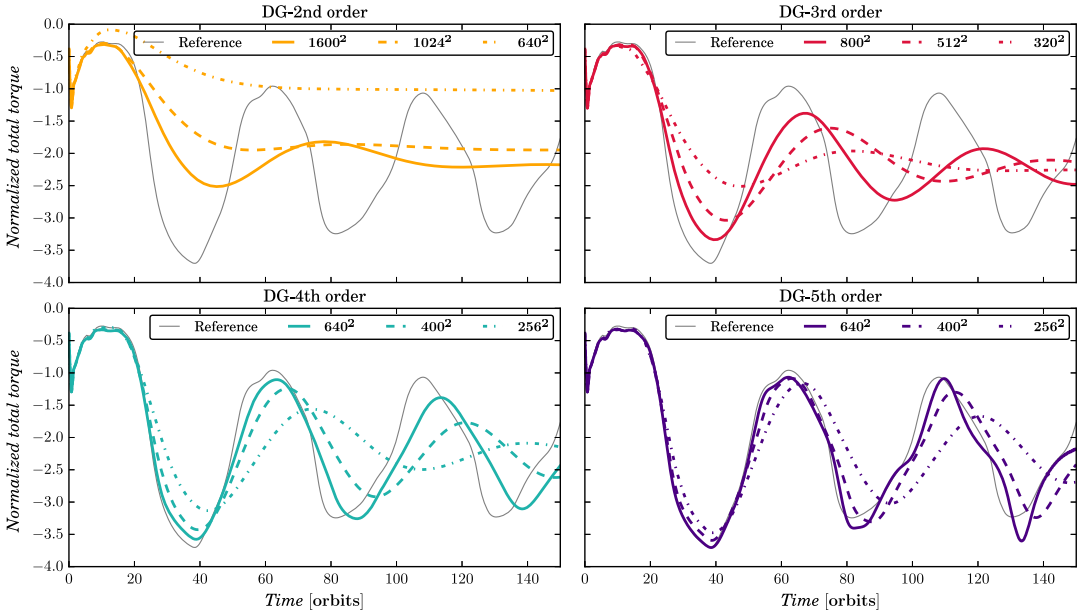

Figure 5. Normalized total torque obtained with the DG code for an adiabatic disc up to 150 orbits, for different orders of the scheme and different resolutions. Each plot corresponds to a given order, and shows the torque evolution for different resolutions. The reference torque is obtained with FARGO3D using orbital advection. No physical viscosity was included in these calculations, and the departure from the serrated behaviour of the torque is exclusively accounted for by numerical diffusion.

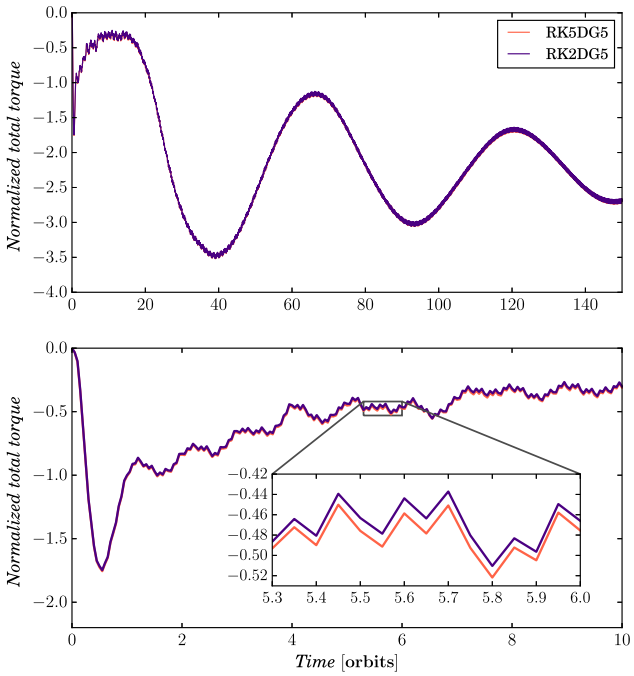

Figure 6. Normalized total torque obtained with the DG code for a fifth spatial order with second and fifth order time integrators.

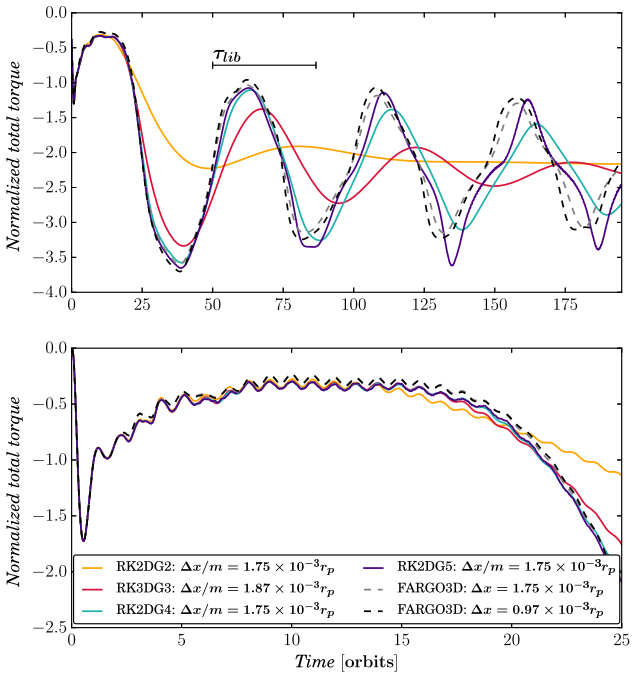

Figure 7. Time dependence of the normalized total torque for an adiabatic disc up to 200 orbits. In solid lines we plot the results for the *DG* code and with dashed lines the ones for FARGO3D with a polar mesh and including the FARGO scheme. We show two different resolutions of the FARGO3D runs, which are nearly undistinguishable, which shows that FARGO3D results are converged.

could be found. Owing to the exquisite sensitivity of the ultimate torque value on the effective viscosity, we believe that our method provides a fair estimate of the scheme's intrinsic viscosity which can be in turn used to assess the scheme's properties in widely different situations, in particular those for which the dominant effect of the scheme properties is a diffusion of vortensity.

The left plot of Fig. 9 shows the advantage, in term of numerical viscosity, of increasing the scheme's order, for a given effective resolution (size of a cell divided by the scheme order). The centre plot shows that FARGO3D with orbital advection performs nearly as *DG* with third order with respect to the resolution at the planet position, while the fourth and fifth order *DG* schemes outperform

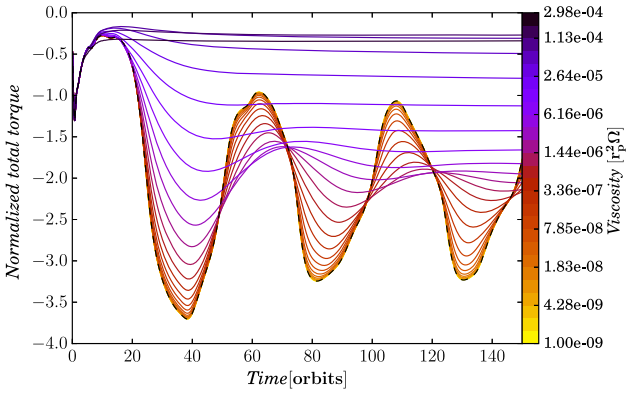

Figure 8. Time dependence of the normalized total torque for adiabatic discs with increasing kinematic viscosity ν. These results are obtained with FARGO3D with a mesh of dimensions 6444×1384 respectively for azimuth and radius. The runs were performed in the corotating frame with orbital advection.

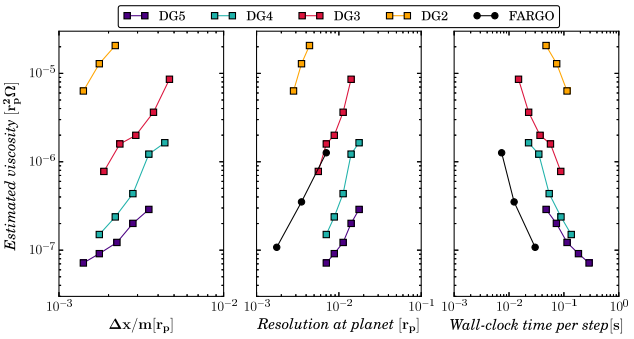

Figure 9. Numerical viscosity of the *DG* schemes inferred from the viscosity palette obtained with FARGO3D. The results are for a second-order Runge–Kutta time integrator for *DG*, and orbital advection and a non-rotating frame for FARGO3D. The left plot shows the inferred viscosity for *DG* as a function of the effective resolution $\Delta x/m$. The centre plot shows the viscosity as a function of the resolution Δx at the planet position. The right plot shows the viscosity as a function of the wall-clock time per step, which shows that FARGO3D is more efficient than DG schemes, at least up to order 5.

FARGO3D, despite the considerably less favourable mesh geometry and the lack of orbital advection. With respect to execution time, the right plot shows that FARGO3D outperforms *DG*, by nearly an order of magnitude.

6 CONCLUSIONS

We have shown the applicability of the DG methods to simulations of planet–disc interactions, being able to obtain negligible numerical viscosities with high-order schemes. We have shown that for a given number of degrees of freedom we reach lower viscosities by increasing the order of the scheme rather than by increasing the resolution. The *DG* code with a Cartesian mesh and a non-rotating frame is able to reproduce the results of FARGO3D, for which we need a polar mesh and either orbital advection or a frame corotating with the planet to properly capture the disc's torque. We note that the effective viscosities of protoplanetary discs may be extremely small. Many observations suggest the existence of vortices, the persistence of which requires a parameter of Shakura–Syunyaev α of at most 10^{-4} (Zhu & Baruteau 2016), which translates in our setup into $\nu = 2.5 \cdot 10^{-7} r_p \Omega_p^2$. Besides, it has been suggested that angular momentum transport driving accretion in protoplanetary discs might not be of viscous nature (Rafikov 2017), which stresses the need for numerical methods with very low numerical viscosity. Our *DG* code is slower than FARGO3D and therefore is probably of little use for single disc setups. Also, being at the present time two-dimensional, it should essentially be regarded as a proof of concept. It nevertheless strongly suggests that DG schemes may be very useful in more complex situations when low-viscosity flows must be captured, such as multiscale simulations of protoplanetary discs and their environment, for which Cartesian AMR are a tool of choice.

REFERENCES

Bai X.-N., Stone J. M., 2013, ApJ, 769, 76
Benítez-Llambay P., Masset F. S., 2016, ApJS, 223, 11
Boxi L., Chao Y., Shusheng C., 2017, Int. J. Comput. Fluid Dyn., 31, 339
Cash J. R., Karp A. H., 1990, ACM Trans. Math. Softw., 16, 201
Cockburn B., Shu C.-W., 1998, J. Comput. Phys., 141, 199
Crida A., Morbidelli A., Masset F., 2006, Icarus, 181, 587
D'Angelo G., Kley W., Henning T., 2003, ApJ, 586, 540
de Val-Borro M. et al., 2006, MNRAS, 370, 529
Duffell P. C., 2016, ApJS, 226, 2
Flaherty K. M., Hughes A. M., Teague R., Simon J. B., Andrews S. M., Wilner D. J., 2018, ApJ, 856, 117
Fung J., Shi J.-M., Chiang E., 2014, ApJ, 782, 88
Gottlieb S., Shu C. W., 1998, Math. Comput., 67, 73
Hennebelle P., Lesur G., Fromang S., 2017, A&A, 599, A86
Hesthaven J. S., Warburton T., 2007, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, 1st edn. Springer Publishing Company, Incorporated, New York
Jiménez M. A., Masset F. S., 2017, MNRAS, 471, 4917
Lesur G., Kunz M. W., Fromang S., 2014, A&A, 566, A56
Lichtenberg T., Schleicher D. R. G., 2015, A&A, 579, A32
Lin D. N. C., Papaloizou J., 1986, ApJ, 307, 395
Liska R., Wendroff B., 2003, Hyperbolic Problems: Theory, Numerics, Applications, Springer, Berlin, Heidelberg, p. 831
Lyra W., Johansen A., Klahr H., Piskunov N., 2009, A&A, 493, 1125
Masset F., 2000, A&AS, 141, 165
Masset F. S., 2001, ApJ, 558, 453
Masset F. S., Casoli J., 2010, ApJ, 723, 1393
Mignone A., Flock M., Stute M., Kolb S. M., Muscianisi G., 2012, A&A, 545, A152
Paardekooper S.-J., Mellema G., 2006, A&A, 450, 1203
Paardekooper S.-J., Baruteau C., Kley W., 2011, MNRAS, 410, 293
Pepliński A., Artymowicz P., Mellema G., 2008, MNRAS, 386, 164
Rafikov R. R., 2017, ApJ, 837, 163
Ruuth S. J., Spiteri R. J., 2002, J. Sci. Comput., 17, 211
Schaal K., Bauer A., Chandrashekar P., Pakmor R., Klingenberg C., Springel V., 2015, MNRAS, 453, 4278
Springel V., 2010a, ARA&A, 48, 391
Springel V., 2010b, MNRAS, 401, 791
Ward W. R., 2007, LPI Contrib., 38, 2289
Yee H. C., Sandham N. D., Djomehri M. J., 1999, J. Comput. Phys., 150, 199
Young M., Ooi A., 2004, in Proceedings of the Fifthteenth Australasian Fluid Mechanics Conference, Behnia, M., School of Aerospace and Mechanical Engineering, Sydney, Australia.
Zhang Q., Shu C.-W., 2004, SIAM J. Numer. Anal., 42, 641
Zhang X., Shu C.-W., 2010, J. Comput. Phys., 229, 8918
Zhu Z., Baruteau C., 2016, MNRAS, 458, 3918
Zhu Z., Stone J. M., Rafikov R. R., Bai X.-n., 2014, ApJ, 785, 122

Table A1. Generic Butcher tableau for k-stage explicit Runge–Kutta method.

0					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
...	...	...	...		
c_k	$a_{k,1}$	$a_{k,2}$	...	$a_{k,k-1}$	
	b_1	b_2	...	b_{k-1}	b_k

Table A2. Runga–Kutta Butcher tableaus for the SSP(2, 2) scheme.

0			
1/2	1/2		
	1/2	1/2	

Table A3. Runga–Kutta Butcher tableau for the SSP(3, 3) schemes.

0			
1	1		
3/4	1/4	1/4	
	1/6	1/6	2/3

APPENDIX A: TIME-STEPPING COEFFICIENTS

To perform the time integration a Runge–Kutta method is used. For the ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}u_h = \mathcal{L}(u),$$

and a suitable initial condition u_h^0, we obtain the solution at t^{n+1}:

$$u_h^{n+1} = u_h^n + h\sum_{i=1}^{k} b_i k_i,$$

where

$$k_i = \mathcal{L}(t^n + c_i \cdot h,\, y_i + h(a_{i,1}k_1 + ... + a_{i,i-1}k_{i-1})).$$

To specify a particular time-stepping method, one needs to specify the number of stages k and the coefficients $a_{i,j}$, b_i, and c_i. This section contains the Butcher tableaus for the different Runge–Kutta time-stepping algorithms. The generic Butcher tableau can be seen in Table A1, and shown in Tables A2, A3, A4, and A5 we have the second-, third-, fourth-, and fifth-order time integration algorithms, respectively.

Table A4. Runga–Kutta Butcher tableau for the SSP-RK scheme RK(4, 5).

0					
0.391 752 227 003 92	0.391 752 227 003 92				
0.586 079 688 967 79	0.217 669 096 338 21	0.368 410 592 629 59			
0.474 542 363 026 87	0.082 692 086 709 50	0.139 958 502 069 99	0.251 891 774 247 38		
0.935 010 631 009 24	0.067 966 283 703 20	0.115 034 698 444 38	0.207 034 898 649 29	0.544 974 750 212 37	
	0.146 811 876 186 61	0.248 482 909 245 56	0.104 258 830 366 50	0.274 438 900 919 60	0.226 007 483 193 95

Table A5. Six-stage Cash–Karp Butcher tableau for fifth-order accuracy (Cash & Karp 1990).

0						
1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	−9/10	6/5			
1	−11/54	5/2	−70/27	35/27		
7/8	1631/55296	175/512	575/13824	44275/110592	253/4096	
	37/378	0	250/621	125/594	0	512/1771

This paper has been typeset from a TEX/LATEX file prepared by the author.